Improving Network Intrusion Detection Performance

An Empirical Evaluation Using Extreme Gradient Boosting (XGBoost) with Recursive Feature Elimination

Gerard Shu Fuhnwi
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
gerard.shufuhnwi@student.montana.edu

Matthew Revelle
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
matthew.revelle@montana.edu

Clemente Izurieta
Gianforte School of Computing
Montana State University
Pacific Northwest National Laboratory
Idaho National Laboratory
Bozeman, MT, USA
clemente.izurieta@montana.edu

***Abstract*—In cybersecurity, Network Intrusion Detection Systems (NIDS) are essential for identifying and preventing malicious activity within computer networks. Machine learning algorithms have been widely applied to NIDS due to their ability to identify complex patterns and anomalies in network traffic. Improvements in the performance of an IDS can be measured by increasing the Matthew Correlation Coefficient (MCC), the reduction of False Alarm Rates (FARs), and the maintenance of up-to-date signatures of the latest attacks to maintain confidentiality, integrity, and availability of services. Integrating machine learning with feature selection for IDSs can help eliminate less important features until the optimal subset of features is achieved, thus improving the NIDS.**

In this research, we propose an approach for NIDS using XGBoost, a popular gradient boosting algorithm, with Recursive Feature Elimination (RFE) feature selection. We used the NSL-KDD dataset, a benchmark dataset for evaluating NIDS, for training and testing. Our empirical results show that XGBoost with RFE outperforms other popular machine learning algorithms for NIDS on this dataset, achieving the highest MCC for detecting NSL-KDD dataset attacks of type DoS, Probe, U2R, and R2L and very high classification time.

***Index Terms*—network intrusion detection; feature selection; RFE ; XGBoost; recursive feature elimination; NSL-KDD.**

I. Introduction

With the increasing reliance on technology in our daily lives, cybersecurity has become a crucial aspect of ensuring confidentiality, integrity, and availability of information. Network Intrusion Detection Systems (NIDS) (an application of anomaly detection [1], which in machine learning, is the process of finding data patterns (outcomes, values, or observations) that deviate from the rest of the other observations or outcomes) are essential for identifying and preventing attacks within computer networks. Computer network attacks come in many forms, including Denial of Service (DoS), Probe, User to Root (U2R), and Remote to User (R2L). Denial of service (DoS) is one of the most common attacks on network resources that make services inaccessible to users [2]. Remote to User (R2L) is another computer attack where attackers send packets to another computer or server over a network without permission. User to Root (U2R) is a third type of attack in which intruders access the network resources as a normal user, and after several attempts, they gain access as a potential root user. Probing is a fourth type of computer attack in which the attackers scan through network devices to check for weaknesses in the network topology and then use them in the future for illegal activities [3]. The most common types of probing attacks are network scans, portsweep, ipsweep, and satan. Traditional rule-based NIDS have proven insufficient in detecting these attacks (DoS, Probe, U2R, and R2L), as they often rely on known attack patterns, so machine learning algorithms have been widely applied to NIDS due to their ability to identify complex patterns and anomalies in network traffic. In recent years, gradient boosting algorithms, such as XGBoost [4], have become increasingly popular in machine learning due to their ability to handle missing values in datasets, the integration of regularization techniques, their optimization for parallel and distributed computing, optimized tree pruning, and a customizable objective function, thereby making it outperform other baseline and state-of-the-art machine learning algorithms [5]. In this paper, we present an approach for NIDS using XGBoost with Recursive Feature Elimination (RFE) for feature selection. RFE is a feature selection technique that recursively eliminates less important features until the optimal subset of features is achieved. The NSL-KDD dataset [6] was used for evaluation, and it contains attacks such as DoS, R2L, U2R, and Probe, a benchmark dataset for evaluating NIDS. Our **goal** can be stated as:

Goal: *Evaluate the effectiveness of XGBoost with RFE in detecting network intrusions and compare its performance to other baseline and state-of-the-art machine learning algorithms.*

Numerous intrusion detection approaches rely on feature selection, but many lack explicit discussions on complexity and time consumption, often rendering them impractical for real-world applications due to potential inefficiencies. This prompts our proposal for a rapid, efficient, and straightforward intrusion detection solution that leverages Recursive Feature Elimination (RFE) to identify the most impactful features, ensuring both speed and effectiveness in real-world scenarios.

Main Contributions: The main contributions of this research include:

- An intrusion detection framework that uses a two-layer ensemble learning (combining RFE with XGBoost) to improve current intrusion detection solutions.
- An expansion of the dataset's feature set by incorporating over 78 new features in the NSL-KDD for the purpose of testing and evaluating the proposed framework.
- Improvements to the overall speed of detection.
- Evidence of model performance improvements through statistical tests.
- Improvements to detection capabilities on selected attack types when compared to other models and benchmarks in related work.

This paper presents experimental results, showing that XGBoost with RFE outperforms other popular machine learning algorithms for NIDS on the NSL-KDD dataset, achieving the highest MCC for detecting NSL-KDD dataset attacks of type DoS, Probe, U2R, and R2L. This approach also achieved high precision, recall, ROC AUC, F1-score values, and low False Alarm Rate (FAR), indicating its effectiveness in detecting network intrusions. Overall, our empirical results demonstrate the potential of XGBoost with RFE for NIDS and provide valuable insights for future research in this field.

The rest of this paper is organized as follows. Section II discusses related work. Section III presents our proposed approach, including phases such as data set description, preprocessing, feature selection, and classification model. The experimental results are presented in Section IV, and Section V concludes the paper describing possible future work.

II. Related Work

Several studies have explored the application of data mining, statistics and machine learning algorithms for network intrusion detection, including gradient boosting algorithms like XGBoost and feature selection techniques like RFE.

Alkasassbeh and Almseidin [3] used three machine learning methods (J48, MLP and Bayes Network classifiers) on the KDD dataset [7] for detecting and classifying attacks such as DoS, R2L, U2R, and Probe, with the J48 classifier achieving the highest accuracy.

In [8], Farnaaz and Jabbar proposed a detection intrusion system using a random forest with feature subset selection method called symmetrical uncertainty. Experimental results were conducted on the NSL-KDD dataset [6]. Empirical results show that the proposed model achieved a low FAR and high recall.

Recently feature selection has been applied to combine machine learning techniques for intrusion detection. A study by Meftah et al. [9] applied RFE and random forests as feature selection methods to the UNSW-NB15 dataset [10], then apply machine learning techniques such as Logistic Regression, Gradient Boost Machine, and Support Vector Machines. Similarly other papers such as; Elmasey et al. [11], Network Harmad et al. [12],Sharma et al. [13], Kunhare et al. [14], Mohammed et al. [15], Saheed et al. [16],Souza et al. [17], Zhang et al. [18], Wang et al. [19], Aljawarneh et al. [20] and Louk et al. [21] both applied feature selection methods together with machine learning techniques for network intrusion detection systems. These papers compare machine learning algorithms with feature selection without the use of statistical methods for choosing their final model and time-consuming issues in their work. To overcome these drawbacks in the models listed in the related works, one can look at the statistical significance tests to compare the performance of machine-learning models and quantify the likelihood of the samples of performance scores being observed, given the assumption that they were drawn from the same distribution and classification time of each machine learning model. There are also other limitations in the related works; for example, lack of good preprocessing of the NSL-KDD dataset for the categorical features such as protocol_type (3 types), service (70 types), flag (11 types), and label (23 types), a lack of evaluation techniques for these machine learning techniques due to the high level of imbalances in the class labels.

Overall, these studies demonstrate the effectiveness of machine learning and feature selection for network intrusion detection. Our study builds on these papers to extend the work of Farnaaz et al. [8], Souza et al. [17], Louk et al. [21] on intrusion detection and the above-mentioned limitations by applying XGBoost with RFE to the NSL-KDD dataset and achieving a high MCC. By combining XGBoost with RFE, additional evaluation techniques for class imbalances, classification time of each attack, and statistical inference, we could identify the essential features of our model and achieve high performance in detecting network intrusions.

III. Proposed Approach: XGBoost with RFE

The proposed approach to test the effectiveness of XGBoost with RFE in a network intrusion detection system has seven steps described herein and shown in Fig. 1.

XGBoost with RFE
Step 1: Read NSL-KDD dataset.
Step 2: Preprocess data.
Step 3: Apply recursive feature elimination.
Step 4: Give XGBoost the select features for training.
Step 5: Find the attack type and classes.
Step 6: Evaluate the model performance using detection

rate, false alarm, Matthews correlation coefficient, F_1 score, the area under the curve of the receiver operating curve, and precision on the KDDTest dataset.
Step 7: Statistical Evaluation

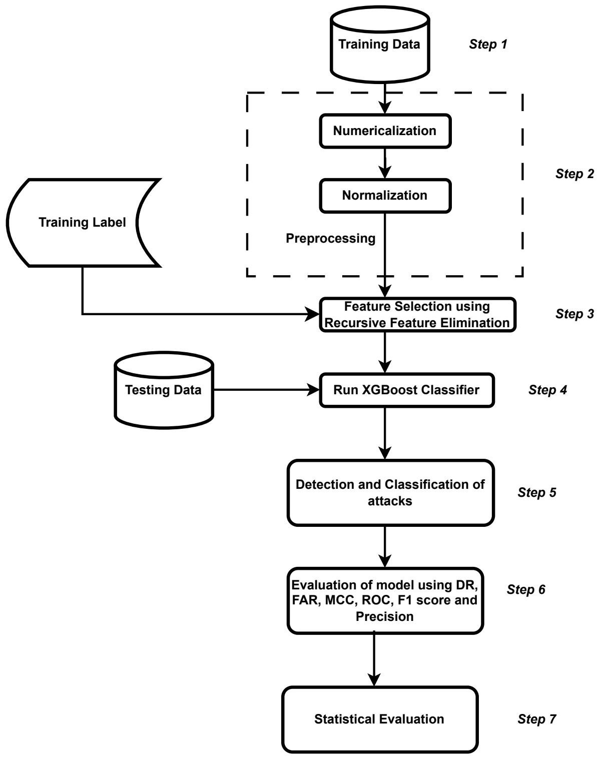

Fig. 1. Flow chart of proposed model approach where the rectangles represent the model's processes, cylinders represent stored data, the trapezoid represents the stored training labels (Normal, DoS, Probe, R2L, and U2L), and the arrows the direction of flow.

A. Dataset Description (step 1)

The NSL-KDD dataset is an improved version of the KDD99 dataset, in which a large amount of data redundancy has been removed [24]. This dataset has the same attributes as the KDD99 having 41 features that are labeled into either normal or one of four attack categories such as DoS, Probe, U2R, and R2. The NSL-KDD dataset repository has two files; KDDTrain.txt and KDDTest.txt. Training is performed on the KDDTrain data, which has 23 attack types, and testing is performed on KDDTest data with 38 (15 additional) attack types, all grouped into the four attack categories. Table I shows the attack types, attack categories (classes), and number of data points per category in the NSL-KDD train and test datasets. The NSL-KDD dataset has 125973 data points in the training dataset and 22544 in the testing dataset.

Our proposed approach is also capable of detecting the 15 unknown attacks in the test dataset.

TABLE I
THE ATTACK CATEGORY(CLASS), THE NUMBER OF RECORDS IN THE NSL-KDD TRAINING AND TESTING DATASETS, AND A SUBSET OF EXAMPLES FOR EACH ATTACK CATEGORY (ATTACK TYPES)

Attack Class	No. of records		Attack Types
	Training	Testing	
Normal	67, 343	9,711	Normal traffic data
DoS	45,927	7, 460	Worm, Land, Smurf, Udpstorm, Teardrop, Pod, Mailbomb, Neptune, Process table, Apache2, Back
Probe	11, 656	2, 421	Ipsweep, Nmap, Satan Portsweep, Mscan, Saint
R2L	995	2, 885	WarezClient, Worm, SnmpGetAttack, WarezMaster, Imap, SnmpGuess, Named, MultiHop, Phf, SPy, Sendmail, Ftp_Write, Xsnoop, Xlock, Guess_Password
U2R	52	67	Buffer_Overflow, SQLattack, Rootkit, Perl, Xterm, LoadModule, Ps, Httptuneel

B. Data Preprocessing (step 2)

The primary goal of data preprocessing is to improve the quality of the data, making it easier to work with and enabling more accurate and efficient results for machine learning algorithms [25].

1) Numericalization: Numericalization converts non-numeric data, such as text or categorical variables, into a numerical format that can be used as input for machine learning algorithms, statistical analysis, or other computational methods. Machine learning algorithms and many statistical methods primarily work with numerical data, so it is essential to represent non-numeric information in a format that these algorithms can understand and process. It is important to note that although the labels of a category are now represented by numbers, the data in it of itself remains categorical and the numbers do not represent ordinal, interval or ratio scales. The NSL-KDD dataset has 41 features, where each feature represents an attack type as described in Section III-A and attack category (class feature). These features are both numeric (38 features) and categorical (3) features. The categorical features are protocol_type (3 types), service (70 types), and flag (11 types) that need to be converted to numeric features. Label encoding is then used to assign each unique type in a categorical variable a distinct integer value. After applying label encoding, one-hot encoding is used in creating binary (0 or 1) features for each unique type in a categorical variable. The attack category (class) feature is also labeled with a numeric type, starting with Normal labeled as 0, DoS labeled as 1, Probe labeled as 2, R2L labeled as 3, and U2R labeled as 4. We further converted the different attack categories to bit form and used 10000 for Normal, 01000 for DoS, 00100 for Probe, 00010 for R2L, and 00001 for R2L. After numericalization, we are left with

122 features and one categorical class feature.

2) Normalization (step 2): Normalization is a technique used to scale numerical features in a dataset to a standard range, typically [0, 1] or [-1, 1]. Normalization aims to bring different features onto a similar scale, making it easier for machine learning algorithms to process the data and reducing the risk of certain features dominating the model due to their larger numerical values. There are several methods for normalization, two of the most common being Min-Max Scaling and Z-score Standardization. In our proposed approach, the Z-score standardization, which scales the value of the entire feature, is used. In Z-score standardization, the average μ of each numerical feature is calculated and then subtracted from the feature value x. The subtracted average and feature x are then divided by the standard deviation σ as shown in the formula:

$$X_i = \frac{x-\mu}{\sigma}$$

C. Feature Selection (step 3)

This is a machine learning process that involves selecting a subset of the most relevant and informative features from a dataset's original set of features. The goal of feature selection is to improve machine learning model performance, interpretability, and generalization of machine learning models by reducing noise, overfitting, and computational complexity. There are three main categories of feature selection techniques; Filter methods, Wrapper methods, and Embedded methods [26]. After numericalization, the NSL-KDD has 122 features, which are not all required for the network intrusion detection system. Essential features are selected using a wrapper feature selection method called recursive feature elimination, which considers the interaction between features and their contribution to the specific model being used. Using recursive feature elimination, the most relevant features in the dataset are identified iteratively, training the classifier model and eliminating the least important features.

After applying recursive feature elimination, 45 out of 122 features are selected for each of the four attack categories. It is important to note that a feature can be selected in more than one attack category.

D. Extreme Gradient Boosting (XGBoost) (step 4)

Extreme Gradient Boosting [27], or XGBoost, is a scalable, high-performance implementation of the gradient boosting algorithm, a popular ensemble learning method. Gradient boosting combines weak learners, typically decision trees, iteratively to create a robust model that minimizes prediction error. XGBoost has several advantages, including speed, parallelization, regularization, handling of missing data, customizable loss functions, ability to handle imbalance classes in the data and built-in cross-validation. The selected features for the different attacks are then applied to the XGBoost classifier. During this process, the XGBoost classifier learns to distinguish between normal traffic and various types of attacks based on the features extracted from the NSL-KDD train and because of the above mentioned advantages, XGBoost with RFE can improve the detection time, MCC and FAR.

E. Detection and Classification (step 5)

Detection and classification of attacks in network intrusion detection systems involve identifying the attack categories on the NSL-KDD Test using the trained XGBoost model.

F. Model Evaluation (step 6)

Once the XGBoost classifier is trained using the NSL-KDDTrain dataset, it is evaluated using NSL-KDDTest dataset. The performance of the XGBoost classifier is evaluated using detection rate (DR), false alarm rate (FAR), Matthews correlation coefficient (MCC), F_1 score, ROC AUC, and precision metrics. These performance metrics are calculated using True Positive (TP), False Negative (FN), False Positive (FP) and True Negative (TN) as shown in the confusion matrix [22] in Table II.

- **Detection Rate (DR):** It is the ratio between the total number of attacks detected by the NIDS to the total number attacks present in the dataset [8] which can be calculated using the formula:

$$DR = \frac{TP}{TP+FN}$$

- **Precision:** This measures the fraction of examples predicted as attacks that turned out to be attacks which can be calculated using the formula:

$$\text{Precision} = \frac{TP}{TP+FP}$$

- **F_1 Score:** It is the harmonic mean of the fraction of examples predicted as attacks that turned out to be attacks (precision). It can also be described as the ratio between the total number of attacks detected by the NIDS to the total number of attacks present in the dataset (detection rate) which can be calculated using the formula:

$$F_1 \text{ Score} = \frac{2*TP}{2*TP+FN+FP}$$

- **False Alarm Rate (FAR):** It is the fraction of non attacks that are misclassified as attacks which can be calculated using the formula:

$$FAR = \frac{FP}{FP+TN}$$

- **Matthews correlation coefficient (MCC):** It is defined as the Pearson product-moment correlation coefficient using the confusion matrix in Table II between the actual attacks and predicted attacks [23] which can be calculated using the formula below. MCC ranges between $[-1,+1]$, where -1 corresponds to the worst overall system performance and 1 corresponds to the best overall system performance. A high MCC score indicates that the binary classifier was able to correctly predict the majority of the attacks and the majority of non-attacks.

$$MCC = \frac{TN * TP - FN * FP}{\sqrt{(FP + TP)(FN + TP)(TN + FP)(TN + FN)}}$$

- **Receiver Operating Characteristic (ROC) Curve:** The Receiver Operating Characteristic (ROC) curve is a graphical representation used to evaluate the performance of binary classification models in machine learning. It is created by plotting the ratio between the total number of attacks detected by the NIDS to the total number of attacks present in the dataset (detection rate) against the fraction of non-attacks that are misclassified as attacks (False Alarm Rate) at various classification threshold levels. The area under the curve (AUC) of the ROC quantifies the overall performance of the classification model. AUC values range from 0 to 1, with a value of 0.5 representing a random classifier and a value of 1 indicating a perfect classifier. A higher AUC value suggests a better-performing classification model.

A good NIDS should have high detection rates, precision, MCC, ROC AUC, F_1 score but low FAR.

TABLE II
CONFUSION MATRIX: A CONTINGENCY CONTAINING FOUR METRICS, TRUE POSITIVE (TP), TRUE NEGATIVE (TN), FALSE POSITIVE (FP), AND FALSE NEGATIVE (FN).

Attack		**Predicted Class**	
		Yes	**No**
Actual class	**Yes**	**TP**	**FN**
	No	**FP**	**TN**

G. Statistical Evaluation (step 7)

We use the two-sample t-test to determine the statistical significance of our proposed approach. The two-sample t-test, also known as the independent samples t-test or unpaired t-test, is a statistical hypothesis test used to compare the means of two independent groups to determine if there is a significant difference between them. The test assumes that the data is normally distributed and the variances of the two groups are equal (although modifications are available if this assumption does not hold). To compare the performance of our proposed approach against the Random Forest and Decision Tree approaches, we used a two-sample t-test to test whether there is a significant difference between the mean performances of DR, FAR, MCC, F_1 score, ROC AUC, and precision. The null hypothesis (H_0) for the two-sample t-test states that there is no significant difference between the mean performances of the two models. In contrast, the alternative hypothesis (H_1) states a significant difference, unlikely to have occurred by chance, between the mean performances of the two models.

IV. EXPERIMENTAL RESULTS

All experiments were performed in Python using default parameters for XGBoost library, Random Forest (Sklearn) and Decision Trees (Sklearn) using Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30GHz processor with 16gb RAM. Training and testing of XGBoost, Random Forest and Decision Tree algorithm were performed on 122 features and the XGBoost model on the 45 selected features from the NSL-KDD dataset. Ten StratifiedKFold cross-validation was adopted during the training and testing of the classifiers to check if the classifiers were not overfitting on both the training and testing. However, the evaluation of the proposed approach was done only on the test dataset by finding the average and standard deviation of the DR, F_1 score, ROC AUC, precision, FAR, and MCC metrics. The performance of our proposed approach is shown in last row of Table IV. It is evident from Table IV that our proposed approach have high MCC, precision, ROC AUC, and low FAR.

To compare the differences of our proposed approach (XGBoost with RFE) against Random Forest and Decision Tree, we executed a two-sample t-test (checks for normality and equal variance assumptions were valid) [28]. Table V compares XGBoost with RFE) against the Random Forest and Decision Tree algorithms on various metrics and attack types with significant values depicted in sky blue. From Table V, comparing XGBoost with RFE against the Random Forest, we can see that the p-values of the test for MCC (for the U2R attack type) and FAR (for the R2L attack type) are significant at $\alpha = 0.05$. Similarly, comparing XGBoost with RFE against the Decision Tree, we can see that the p-values of the test for MCC (U2R attack type), FAR (R2L attack type), ROCAUC (for the U2R attack type), and precision (U2R attack type) are significant at $\alpha = 0.05$. Also, from Table V comparing XGBoost with RFE against the XGBoost, we can see that the p-value of the test for FAR (R2L attack type) is significant at $\alpha = 0.05$. We can statistically confirm that our proposed approach (XGBoost with RFE) outperforms Random Forest (on MCC and FAR) and Decision Tree (on MCC, FAR, ROC AUC, and precision).

The last row of Table III shows the approximate run time for the intrusion to be classified as DoS, Probe, R2L, and U2R. From this, it can be identified that the classification time displayed is linear, which proves our proposed approach's scalability. Overall, the proposed approach has a classification time of 0.009 milliseconds for Dos, 0.020 milliseconds for Probe, 0.024 milliseconds for R2L, and 0.655 milliseconds for U2R, which all the models in the related works fail to mention. It is worth noting that the run time of Random Forest and Decision Tree looks better than our proposed approach because there are inherently parallel algorithms compared to XGBoost, which is a sequential algorithm, and since all the run times in III are approximately linear, XGBoost with REF is significantly excellent at classifying Dos, Probe, R2L, and U2R.

Hence, from the results shown in our experiments, we can conclude that XGBoost with RFE for NIDS is valuable in cybersecurity.

A. Discussion of Results

In this paper, we investigated the effectiveness of using XGBoost with recursive feature elimination (RFE) for network intrusion detection. Our primary goal was to empirically determine whether combining XGBoost and RFE could yield better detection rates than other methods while reducing false positives and negatives. We employed a comprehensive methodology, analyzing a diverse network traffic dataset, and evaluated our model using several performance metrics, variability of the performance metrics (standard deviation), and classification time of the attack types. Our results indicate that the XGBoost model with RFE achieved a high level of performance in terms of precision, DR, F1 score, MCC, and ROC AUC. This performance indicates that the model can accurately detect network intrusions while maintaining a low false alarms rate (FAR), classification time and low variability (standard deviation) for all the performance metrics. Our approach improved detection capabilities on selected attack types when compared to other models and benchmarks in related work. The RFE process identified several key features highly relevant to network intrusion detection. These features align with our expectations and prior research [15], [17] and [19], confirming the importance of specific traffic characteristics in detecting malicious activities. The combination of XGBoost and RFE not only improved the model's performance but also improves the detection speed and reduces the complexity of the model by eliminating redundant and irrelevant features as compared to related works that focus on network intrusion detection as shown in Table VI.

The practical implications of our findings are significant for the field of network intrusion detection. The improved detection rates and speed offered by our approach can help security practitioners identify and respond to cyber threats more effectively. Additionally, reducing false positives and negatives can minimize the operational overhead of manually investigating false alarms. Furthermore, our approach demonstrates potential scalability and adaptability for different network environments and evolving cybersecurity threats.

Despite these promising results, our study has some limitations and threats to validity:

1) The dataset does not fully capture the diversity of network traffic patterns and evolving attack techniques. Future work could benefit from more recent and diverse datasets to validate our approach. This is a threat to the external validity of our work.
2) The XGBoost algorithm, while highly effective, may be prone to overfitting and is often difficult to interpret. Alternative methods, such as partial dependency plots (PDPs), SHAP (SHapley Additive explanation), and LIME (Local Additive Interpretable model-agnostic explanations), could be investigated to address these issues.

V. Conclusion and Future work

This paper presents XGBoost with RFE to detect four types of attacks DoS, Probe, U2R, and R2L. We adopted ten stratified k-fold cross-validations. Our proposed approach is evaluated using the NSL-KDD dataset. We compared our proposed method with XGBoost, Random Forest, and Decision Tree using the following metrics: DR, F_1 score, ROC AUC, precision, FAR, and MCC. Because of the usage of feature selection, the computational cost decreases, and our experimental results indicate that our proposed approach increases the DR, F1 score, ROC AUC, precision, MCC, and decreases FAR, classification time, variability within all the performance metrics for all of attacks. We equally compared our proposed method against Random Forest and Decision Tree for selected attack types using a two-sample t-test, and found that our proposed approach (with fewer features) is promising. Moreover, this model is compared with related works that focus on network intrusion detection. The comparison results in Table VI showed that XGBoost with RFE has better performance, and less classification time.

For future work, we will experiment with deep learning approaches like GANs and autoencoders since they are capable of handling data of higher dimensions and to address computationally expensive recursive feature elimination.

TABLE III
Approximate Classification Time on NSL-KDD Test.

Classifier	Attack type	Time (Seconds)
XGBoost (122 Features)	DoS	170.3
	Probe	109.9
	R2L	149.1
	U2R	85.3
Random Forest (122 Features)	DoS	57.1
	Probe	36.4
	R2L	44.1
	U2R	37.4
Decision Tree (122 Features)	DoS	8.4
	Probe	3.8
	R2L	6.0
	U2R	3.3
XGBoost (45 Features)	**DoS**	**68.1**
	Probe	**49.6**
	R2L	**68.8**
	U2R	**43.9**

TABLE IV
Performance Measure together with standard deviation (STD) on NSL-KDD Test.

Classifier	**Attack type**	**DR ± STD**	**F$_1$ Score ± STD**	**ROC AUC ± STD**	**Precision ± STD**	**FAR ± STD**	**MCC ± STD**
XGBoost (122 Features)	DoS	0.99745 ± 0.00423	0.99745 ± 0.00294	0.99997 ± 0.00007	0.99812 ± 0.00215	0.00144± 0.00002	0.99609±0.00008
	Probe	0.99463 ± 0.00617	0.99547 ± 0.00481	0.99988± 0.00021	0.99633 ± 0.00426	0.00123± 0.00002	0.99096±0.00002
	R2L	0.97792 ± 0.01218	0.97861 ± 0.01006	0.99914 ± 0.00079	0.97938 ± 0.01088	0.00916± 0.00002	0.95723±0.00001
	U2R	0.90217± 0.13172	0.92916± 0.09870	0.99931 ± 0.00150	0.96790 ± 0.10148	0.00041±0.00004	0.86540± 0.00004
Random Forest (122 Features)	DoS	0.99705 ± 0.00335	0.99785 ± 0.00206	0.99989 ± 0.00054	0.99866 ± 0.00240	0.00103± 0.00003	0.99621±0.00009
	Probe	0.99365 ± 0.00588	0.99495 ± 0.00441	0.99986± 0.00026	0.99628 ± 0.00419	0.00113± 0.00001	0.98992±0.00002
	R2L	0.97310 ± 0.01232	0.97419 ± 0.01065	0.99855 ± 0.00096	0.97532 ± 0.00974	0.01081± 0.00002	0.94841±0.00002
	U2R	0.88064± 0.13087	0.90988± 0.09733	0.99943 ± 0.00162	0.95453 ± 0.11931	0.00062±0.00004	0.82417± 0.00002
Decision Tree (122 Features)	DoS	0.99705 ± 0.00506	0.99585 ± 0.00368	0.99647 ± 0.00328	0.99466 ± 0.00505	0.00412± 0.00002	0.99266 ± 0.00002
	Probe	0.99257 ± 0.00617	0.99303± 0.00645	0.99257± 0.00617	0.99349 ± 0.00695	0.00247± 0.00002	0.98606± 0.00003
	R2L	0.96970± 0.01318	0.97052 ± 0.01466	0.97325 ± 0.01259	0.97139 ± 0.01720	0.01277± 0.00001	0.94101±0.00003
	U2R	0.90979± 0.09227	0.89307± 0.07719	0.90979 ± 0.09227	0.88295 ± 0.11079	0.00185±0.00001	0.78490± 0.00003
XGBoost with RFE (45 Features.)	DoS	0.99732 ± 0.00398	0.99725 ± 0.00243	0.99997 ± 0.00008	0.99719 ± 0.00279	0.00216± 0.00002	0.99514±0.00002
	Probe	0.99443± 0.00594	0.99535± 0.00493	0.99985± 0.00022	0.99628 ± 0.00496	0.00123±0.00003	0.99070±0.00003
	R2L	0.97606± 0.00899	0.97670 ± 0.00711	0.99893 ± 0.00130	0.97738 ± 0.00768	0.00009± 0.00007	0.95341±0.00002
	U2R	0.89633± 0.12839	0.93216± 0.09260	0.99875 ± 0.00358	0.98095 ± 0.07407	0.00021± 0.00003	0.87232±0.00003

TABLE V
Results of two-sample t-test
comparing our proposed approach (XGBoost with RFE) against Decision Tree, Random Forest, and XGBoost

Comparison	**Evaluation metric (Attack Type)**	**Test Statistic**	**Hypothesis** ($\alpha = 0.05$)	**p-value**
XGBoost with RFE vs. Decision Tree with significant values highlighted in table IV	**MCC (U2R)**	**18.93**	**Rejected**	**<0.001**
	FAR (R2L)	**134.65**	**Rejected**	**< 0.001**
	ROC AUC (U2R)	**28.42**	**Rejected**	**< 0.001**
	Precision (U2R)	**2.3**	**Rejected**	**0.031941**
XGBoost with RFE vs. Random Forest with significant values highlighted in table IV	**MCC (U2R)**	**3.72**	**Rejected**	**0.00154**
	FAR (R2L)	**37.99**	**Rejected**	**< 0.001**
XGBoost with RFE vs. XGBoost with significant values highlighted in table IV	**FAR (R2L)**	**393.98**	**Rejected**	**< 0.001**

TABLE VI
Comparison With Other Network Intrusion Detection Methods.

Method	**Attack type**	**FAR**	**ROC AUC**	**MCC**	**Speed**	**Statistical Test**	**Complexity**
Farnaaz et al. [8]	DoS	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	Probe	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	R2L	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
	U2R	Low	Not Mentioned	High	Not Mentioned	Not Mentioned	Medium
Souza et al. [17]	DoS	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	Probe	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	R2L	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
	U2R	Not Mentioned	High	Not Mentioned	Not Mentioned	Not Mentioned	Medium
Louk et al. [21]	DoS	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
	Probe	Not Mentioned	Not Mentioned	High	Not mentioned	Included	Not Mentioned
	R2L	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
	U2R	Not Mentioned	Not Mentioned	High	Not Mentioned	Included	Not Mentioned
Proposed Approach	DoS	Very Low	High	High	Very High	Included	Medium
	Probe	Very Low	High	High	Very High	Included	Medium
	R2L	Very Low	High	High	Very High	Included	Medium
	U2R	Very Low	High	High	Very High	Included	Medium

References

[1] G.S. Fuhnwi, J.O.Agbaje, K.Oshinubi,and O.J.Peter (2023), “An Empirical Study on Anomaly Detection Using Density-based and Representative-based Clustering Algorithms,” Journal of the Nigerian Society of Physical Sciences, pp. 1364–1364, 2023.

[2] S. Paliwal and R.Gupta, “Denial-of-service, probing & remote to user (R2L) attack detection using genetic algorithm,”International Journal of Computer Applications, Citeseer, Dec.2012, vol.60, pp. 57–62

[3] M.Alkasassbeh and M.Almseidin, “Machine learning methods for network intrusion detection,”arXiv preprint arXiv:1809.02610, 2018.

[4] T.Chen et al., “Xgboost: extreme gradient boosting,”R package version 0.4-2,2015, vol.1, pp. 1–4.

[5] B.Mahesh, “Machine learning algorithms-a review,” International Journal of Science and Research (IJSR).[Internet], 2020, vol.9, pp. 381–386.

[6] NSL-KDD, “Canadian Institute for Cybersecurity", 2014.

[7] P.Bhoria and K.Garg, “Determining feature set of DOS attacks,” International Journal of Advanced Research in Computer Science and Software Engineering, 2013, vol.3, pp. 875–878.

[8] N.Farnaaz and M.Jabbar, “Random forest modeling for network intrusion detection system,” Procedia Computer Science, Elsevier, Jan.2016, vol.89, pp. 213–217.

[9] S.Meftah, T.Rachidi, and N.Assem, “Network based intrusion detection using the UNSW-NB15 dataset,” International Journal of Computing and Digital Systems, University of Bahrain, Sep.2019, vol.8, pp. 478–487.

[10] N.Moustafa and J.Slay, “The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems,” 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS), IEEE, 2015, pp. 25–31.

[11] W.Elmasry, A.Akbulut, and A.H.Zaim, “Evolving deep learning architectures for network intrusion detection using a double PSO metaheuristic,” Computer Networks, Elsservier, 2020, vol.168, pp. 107042.

[12] T.Hamed, R.Dara, and S.C.Kremer,“Network intrusion detection system based on recursive feature addition and bigram technique,” Computers & Security, Elsevier, 2018, pp. 137–155.

[13] N.V.Sharma and N.S.Yadav, “An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers,” Microprocessors and Microsystems, Elsevier, Sep.2021, vol.85, pp. 104–293.

[14] N.Kunhare, R.Tiwari, and J.Dhar, “Particle swarm optimization and feature selection for intrusion detection system,”Sādhanā, Springer, 2020, vol.45, pp. 1–14.

[15] B.Mohammed and E.K.Gbashi,“Intrusion detection system for NSL-KDD dataset based on deep learning and recursive feature elimination,” Engineering and Technology Journal, University of Technology-Iraq, 2021, vol.39, pp. 1069–1079.

[16] K.Y.Saheed, M.O.Arowolo, and A.U.Tosho, “An Efficient Hybridization of K-Means and Genetic Algorithm Based on Support Vector Machine for Cyber Intrusion Detection System,” International Journal on Electrical Engineering and Informatics, 2022, vol.14, pp. 426–442.

[17] J.D.S.WS and B.Parvathavarthini,“Machine learning based intrusion detection framework using recursive feature elimination method,” 2020 International Conference on System, Computation, Automation and Networking (ICSCAN), July.2020, pp. 1–4, IEEE.

[18] H.Zhang, Y.Li, Z.Lv, A.K.Sangaiah, and T.Huang, “A real-time and ubiquitous network attack detection based on deep belief network and support vector machine,” IEEE/CAA Journal of Automatica Sinica, IEEE, 2020, vol.7, pp. 790–799.

[19] J.Wang, C.Liu, X.Shu, H.Jiang, X.Yu, J.Wang, and W.Wang, “Network intrusion detection based on xgboost model improved by quantum-behaved particle swarm optimization,” IEEE Sustainable Power and Energy Conference (iSPEC), IEEE, 2019, pp. 1879–1884.

[20] S.Aljawarneh, M.Aldwairi, and M.B.Yassein, “Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model,” Journal of Computational Science, Elsevier, 2018, pp. 152–160.

[21] M.H.Louk and B.A.Tama, “Dual-IDS: A bagging-based gradient boosting decision tree model for network anomaly intrusion detection system,” Expert Systems with Applications, Elsevier, 2023, vol.213, pp. 119030.

[22] E.Beauxis-Aussalet and L.Hardman, “IEEE Conference on Visual Analytics Science and Technology (VAST)-Poster Proceedings,”2014, pp. 1–2.

[23] D.Chicco and G.Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” BMC genomics, Springer, 2020, vol.21, pp. 1–13.

[24] M.Tavallaee, E.Bagheri, W.Lu, and A.A.Ghorbani, Ali A, “A detailed analysis of the KDD CUP 99 data set,” IEEE symposium on computational intelligence for security and defense applications, 2009, pp. 1–16.

[25] T.M.Mitchell et al., “Machine learning,” McGraw-hill New York, 2007, vol.1.

[26] Y.Saeys, I.Inza, and P.Larranaga, “A review of feature selection techniques in bioinformatics,” BMC genomics, Oxford University Press, 2007, vol.23, pp. 2507–2517.

[27] T.Chen and C.Guestrin, “Xgboost: A scalable tree boosting system,” Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794.

[28] C.A.Boneau, “A comparison of the power of the U and t tests.,” Psychological Review, American Psychological Association, 1962, vol.69, pp. 246.